# //字符串倒过来

j=strlen(a)-1;//之后就用j了

for(i=0;i<=j;i++)//或者for(i=0;i<strlen(a);i++)

{

b[i]=a[j-i];

}

b[i]=0;

# //将一排单词中的每个单词存到数组中

## 法一：

char a[10][11]={0},b[100],cnt=0;//没有给数组a初始化就要出错。 //二维数组a也是字符类型

int i,j=0,k=0;

gets(b);

for(i=0;b[i]!=0;i++)

{

if(b[i]==' ')

{

j++;

k=0;

cnt++;//cnt在这里是下标（cnt最大值就是最后一个a[j]），打印的时候看一下。

continue;

}

a[j][k]=b[i];

k++;

}

## 法二：包含头文件 string.h或者 cstring。From说反话（PTA乙中文版）

char \*arr2[40];//指针数组代替原来的二维数组。

arr2[0]=strtok(s," "); //第一个截下来的字符串要装起。

i=0;

while(arr2[i]!=NULL)

{

i++;

arr2[i]=strtok(NULL," ");

}

cnt=--i;

# //当阶乘值 前面有0不需要 中间及后面有零不能省 的处理办法。：变动i。

eg. for(i=0;c[i]==0;i++)

if(c[i]!=0) break; //此时的i就是不为0的,接下来打印就是了。

# //处理 用整形数组装 很长数据的方法。（下面例子来自高精度加法）结构：for+while

int flag=0,s;

for(i=0;i<=la-1||i<=lb-1;i++)//s装加起来的值（未进位时的总值），c[i]是要得到的数组，flag是进位操作。//la,lb分别为整形数组长度（数组最大下标加1，初始化都为1）

{ //for循环就把末尾余数保留到c[i]，其他的往后面堆，堆到出循环。

s=a2[i]+b2[i]+flag; //这里的i。。。就是数组C的长度+1。

c[i]=s%10;

flag=s/10;

}//end of for

while(flag!=0)//堆到最后一位，万一出现了进位了，怎么办。

{

c[i]=flag%10;

flag=flag/10;

i++;

}

//输出夹杂这整形数据与字符：printf("+%d",);之类的

# //一个数组b1[k][]装a[][]中不同种类的字符串（用到一个判断函数），一个整形数组b2[k]装对应字符串在a[][]中出现的次数。（色盲的民主）

for(i=0;i<n;i++)//将单词种类存入数组b1。

{

if(f(a[i],b1,n)==1)//函数f判断字符串a[i]是否已经存入b1[][]，是的话返回1.

{

strcpy(b1[k],a[i]);

k++;

}

}

//找对应的个数：将数组b1作为外层循环，找每种字符串在a中出现的个数。

for(i=0;i<k;i++)

for(j=0;j<n;j++)

{

if(strcmp(b1[i],a[j])==0)

{

b2[i]++;

}

}

# //一个数组b1[k][]装字符串，一个数组b2[k]装对应字符串在a[][]中出现的次数。（色盲的民主）

for(i=0;i<n;i++)//将单词种类存入数组b1,再将数组b1作为外层循环，找个数。

{

if(f(a[i],b1,n)==1)//函数f看字符串a[i]是否已经存入b1[][]，是的话返回1. ，学到二维数组整个调用的化传入b1(二维数组的首地址

{ 一维数组调用传首地址a[i];

strcpy(b1[k],a[i]);

k++;

}

}

//找对应的个数。

for(i=0;i<k;i++)

for(j=0;j<n;j++)

{

if(strcmp(b1[i],a[j])==0)

{

b2[i]++;

}

}

# //实现二维数组表示有几个字符串时，将字符串内部排序。

//3层，最外层k决定往下移动，中间的两层for来排序。

for(k=0;k<m;k++)//k是考试次数，i,j是成绩

{for(i=0;i<n-1;i++)//n是人数，也是排名的限制。

{g=i;

for(j=i+1;j<n;j++)

{

if(chengji[k][g]<chengji[k][j])

g=j;

}

if(i!=g)//应该是i！=g

{

t=chengji[k][g];

chengji[k][g]=chengji[k][i];//应该是[k][j]而不是【i][j

chengji[k][i]=t;

}

}

}//外层循坏

# //知道矩阵元素个数(n)，列数(r)，求 排数（cengshu) [排队打水问题】（转换为下标要-1）

if(n/r==n\*1.0/r)

cengshu= n/r;

else

cengshu= n/r+1;

法2.（根据班级排名） cengshu-1=(n-1)/r,（让总个数由0到n-1编号），这样计算层数或者下标更方便。

# //求最长非递减元素个数（阮小二买彩票）

用 arr[N],dp[N]. dp[N]初始化为一，存对应arr[]位置最长非递减元素个数

实现结构 ： for(i=0;i<n;i++)

for(j=0;j<i;j++)

{

if(arr[i]>=arr[j])

{

dp[i]=fmax(dp[i],dp[j]+1);

}

}

# //整形数组，从1开始(a[0]=1)，然后 乘以很大的数字，改编自阶乘计算。

网址：<https://pintia.cn/problem-sets/1215109992850866176/problems/1216186761255264256>

算法适用于：1\*2\*3\*…\*100\*…\*99999999 int型号能装下。 不能： 12345678999 \* 132412;

a[0]=1;//此时个位已经在第一位，往后走就是高位。

length=1;//length就是要在最高位的下标再加一。

for(i=0;i<k;i++)

{

for(j=0;j<length;j++)//先把乘的，过一遍，直到j为length

{

s=a[j]\*b[i]+flag; //b[i]直接是单个的乘数(并非某个位数eg.123 或 6789 或 9823) 而 a[j]就是位数。

a[j]=s%10;

flag=s/10;

}

while(flag!=0)

{

a[length]+=flag%10;

flag=flag/10;

length++;

}

}

## 万一原来的字符串末尾有0？

没关系，因为 是int[] 而不是int 所以0不会被吃掉。

## length放在哪里？

整体结构是 for{ for{ length} while(flag){length} }

# //leetcode字符串相乘(大整数乘法，竖式乘法)

char \* multiply(char \* num1, char \* num2){

    int len1=strlen(num1);

    int len2=strlen(num2);

    if(num1[0]=='0') return num1;

    if(num2[0]=='0') return num2;

    int totallen=len1+len2;//最大位数，这是数学常理

    int i,j;

    int \*value=(int\*)malloc(sizeof(int)\*(totallen+1));

    char \*ans=(char\*)malloc(sizeof(char)\*(totallen+1));

    memset(value,0,sizeof(int)\*(totallen+1));

    memset(ans,0,sizeof(char)\*(totallen+1));

    for(i=len1-1;i>=0;i--){

        for(j=len2-1;j>=0;j--){

            value[i+j+1]+=(num1[i]-48)\*(num2[j]-48);//巧妙地i+j与totallen的关系

        }

    }

    //处理进位

    int s,flag=0;

    for(i=totallen-1;i>=0;i--){

        s=value[i]+flag;

        value[i]=s%10;

        flag=s/10;

    }

    //装入字符串

    i=0;

    while(value[i]==0 && i<totallen){//万一高位有0，那么忽略最高位的0

        i++;

    }

    for(i,j=0;i<totallen;i++){

        ans[j++]=value[i]+48;

    }

    ans[j]=0;

    return ans;

}

## 巧妙

* int totallen=len1+len2;//最大位数，这是数学常理
* for(i=len1-1;i>=0;i--){//直接从最后面开始，不用辅助数组了

        for(j=len2-1;j>=0;j--){

            value[i+j+1]+=(num1[i]-48)\*(num2[j]-48);//巧妙地i+j与totallen的关系

        }

}

# //有许多排数据，每排数据中不止有一个空格。升级一排单词存二维字符数组的想法。（来自 错误票据）。

将每个有效数据存入二维字符数组。

k=0;

for(i=0;i<n;i++)

{

m=0; //换行时，要开拓个b[k]来装下一排的字符，不然的话要等到下一排的'空格'出现才开拓b【k】，会发生数据覆盖现象。

cnt++;

k++;

for(j=0;a[i][j]!=0;j++)

{

if(a[i][j]==' '&& a[i][j+1]!=0 && a[i][j+1]==' ')

{

continue;

}

if(a[i][j]==' '&& a[i][j+1]!=0 && a[i][j+1]!=' ')

{

k++;

m=0;

cnt++;

continue;

}

b[k][m++]=a[i][j];

}

}

## 可以换成strtok(NULL,” “);

# //atoi使用场景将二维字符数组转化为一维整形变量：（来自 错误票据）

for(i=0;i<=cnt;i++)

{

c[i]=atoi(b[i]);

}

# //函数返回一个新数组的首地址，改变传入数组的顺序，知道新数组的大小。（来自错误票据）

int \*Sort(int a[],int n,int \*returnSize)

{

int i,j,k,t;

int \*b;

b=(int\*)malloc(sizeof(int)\*2);

for(i=0;i<n-1;i++)

{k=i;

for(j=i+1;j<n;j++)

if(a[k]>a[j])

k=j;

if(k!=i)

{

t=a[i];

a[i]=a[k];

a[k]=t;

}

}

for(i=0;i<n-1;i++)

{

if(a[i]+2==a[i+1])

{

b[0]=a[i]+1;

}

if(a[i]==a[i+1])

{

b[1]=a[i];

}

}

\*returnSize=2;

return b;

}

# //大整形数组的相加(封装到函数里面得)

//2个整形数组相加得到新的整形数组，用函数void fJia(int \*a,int \*len1,int \*b,int \*len2,int\*c,int \*len3)//c=a+b实现。

//实现整形数组间的赋值操作，用fClean（） 与 Fuzhi（）来实现

int l1=1,l2=1,l3=1;（全局变量，待会儿还是传地址）

void fClean(int \*a,int \*len)//清除掉整形数组所有的值。

{

int i;

for(i=\*len-1;i>=0;i--)

{

a[i]=0;

}

\*len=1;

}

void fJia(int \*a,int \*len1,int \*b,int \*len2,int\*c,int \*len3)//c=a+b

{

int i=\*len3,flag=0,s;

for(i=0;i<=\*len1-1 || i<=\*len2-1;i++)//s装加起来的值（未进位时），c[i]是要得到的数组，flag是进位操作。//len1,len2分别为整形数组长度

{ //for循环就把末尾余数保留到c[i]，其他的往后面堆，堆到出循环。

s=a[i]+b[i]+flag; //这里的i。。。是数组的长度，即（最大下标＋1）

c[i]=s%10;

flag=s/10;

}

while(flag!=0)//堆到最后一位，万一出现了进位了，怎么办。

{

c[i]=flag%10;

flag=flag/10;

i++;

}

\*(len3)=i;

}

void Fuzhi(int \*a,int \*l1,int \*b,int \*l2)//b复制给a

{

int i;

for(i=\*l2-1;i>=0;i--)

{

a[i]=b[i];

}

\*l1=\*l2;

}

void fcount(int n)//放在main里面

{

int i,f1[10000],f2[10000],f3[10000];

f1[0]=1;

f2[0]=1;

if(n==1 || n==0)

{if(n==1)

printf("1");

else printf("0");

}

else

{

for(i=2;i<=n;i++)

{

//f3=f2+f1;

fClean(f3,&l3);

fJia(f1,&l1,f2,&l2,f3,&l3);

//f3=f2+f1;

//f1=f2;

fClean(f1,&l1);

Fuzhi(f1,&l1,f2,&l2);

//f1=f2;

//f2=f3;

fClean(f2,&l2);

Fuzhi(f2,&l2,f3,&l3);

//f2=f3;

}

for(i=l3-1;i>=0;i--)

{

if(f3[i]!=0) break;

}

for(i;i>=0;i--)

{

printf("%d",f3[i]);

}

}

}

# //leetcode字符串相加(字符串化大整形数组)

string addStrings(string num1, string num2) {

vector<int> arr1((num1.length()+num2.length()),0);

vector<int> arr2((num1.length()+num2.length()),0);

vector<int> arr((num1.length()+num2.length()),0);

string ans;

int i,j;

int len1=num1.length(),len2=num2.length();

for(i=0;i<len1;i++){

arr1[i]=num1[len1-1-i]-48;

}

for(j=0;j<len2;j++){

arr2[j]=num2[len2-1-j]-48;

}

int flag=0,sum,length;

for(length=0;length<len1 || length<len2;length++){//len1、len2为长度(最大下标+1)

sum=arr1[length]+arr2[length]+flag;

arr[length]=sum%10;

//flag+=sum/10;//错!

flag=sum/10;

}

while(flag!=0){

arr[length]=flag%10;

length++;

flag=flag/10;

}

ans.resize(len1+len2);

//如果末尾有0,还要弄出去。

for(i=length-1;i>=0;i--){

ans[length-1-i]=arr[i]+48;

}

return ans;

}

# //Z字形 用到goto 语句 leetcode

将一个字符串分别存入多个字符串，这些字符串的个数不相等。

方法：用到strncat（arr[i],&s[i],1）从s[i]开始，录入1个（即s[i]）。

用到goto 语句，上下摆动。

具体代码：

void f(char \*s,int rownum)

{

char arr[rownum][10]={0};

int i=0,j=0;

RE: for(i;s[i]!=0;i++)

{

strncat(arr[j],&s[i],1);

j++;

if(j==rownum)//表示最后一排已经赋值

{

j-=2;

i++;

break;

}

}

for(i;s[i]!=0;i++)

{

strncat(arr[j],&s[i],1);

j--;

if(j==-1)

{

j+=2;

i++;

goto RE;

}

}

for(i=0;i<rownum;i++)

printf("%s\n",arr[i]);

}

# //双重for遍历两条链表，并且建立出新链表。from：多项式相加和相乘

pr1=L1->next;

pr2=L2->next;

L3=(pd)malloc(sizeof(struct DATA));

pr3=L3;

for(pr2=L2->next; pr2!=NULL; pr2=pr2->next)//以下面那排为基准。3个

{

for(pr1=L1->next; pr1!=NULL; pr1=pr1->next)//4个一循环。//我知道了，再次进入循环，不是从L2的头开始，而是NULL1！！

{

p=(pd)malloc(sizeof(struct DATA));

p->ceo=pr1->ceo \* pr2->ceo;

p->ex=pr1->ex + pr2->ex;

pr3->next=p;//尾巴

pr3=pr3->next;//本身

}

}

# //将一维字符数组 12+2+(3\*4)+49 用二维数组装起来。 from:多项式求值

eg:【12】【+】【2】【+】【3】【\*】【4】【+】【49】

for(i=0;s[i]!=0;i++)

{

if(s[i]>='0' && s[i]<='9')//装数字

{

arr[j][k]=s[i];

k++;

continue;

}

if(i!=0 && s[i-1]>='0' && s[i-1]<='9') j++; //关键得一批：如果上一位是 数字，那么j要加一位；如果上一位是 符号，那么j不加（因为下面的j加了）

arr[j][0]=s[i];//装字符

j++;

k=0;

}